

DOCUMENT RESUME

ED 225 532

IR 010 536

AUTHOR Fisher, Patience; And Others
 TITLE Computer Programming: BASIC.
 INSTITUTION Lincoln Public Schools, Nebr.
 PUB DATE Jun 80
 NOTE 60p.
 PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052)

EDRS PRICE MF01/PC03 Plus Postage.
 DESCRIPTORS Behavioral Objectives; Computer Graphics; *Computer Literacy; *Computer Programs; Input Output Devices; Instructional Materials; *Microcomputers; *Programing; Secondary Education; Student Attitudes

IDENTIFIERS *BASIC Programing Language; Lincoln Public Schools NE

ABSTRACT

This guide was prepared to help teachers of the Lincoln Public School's introductory computer programming course in BASIC to make the necessary adjustments for changes made in the course since the purchase of microcomputers and such peripheral devices as television monitors and disk drives, and the addition of graphics. Intended to teach a programming language and to give students an opportunity to apply the language to solving problems, the course is limited to students enrolled in a mathematics course at or above the level of Algebra 3-4. The guide first lists 125 course goals which focus on computer use, problem solving, debugging, attitudes, values and motivation, input-output devices, and data. BASIC programming commands and concepts which should be covered in the course are listed and directions are provided for initializing a microcomputer diskette. System (job control), illustrating, editing, BASIC, and disk operating system (DOS) commands are explained, and directions for using them are given. Remaining chapters discuss low and high resolution graphics, and use of the TRACE Command to see the path of execution in a program. Additional resources listed include 27 supplemental books, 7 periodicals, an educational television series, and a bibliography. (LMM)

 * Reproductions supplied by EDRS are the best that can be made *
 * from the original document. *

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

This document has been reproduced as
received from the person or organization
originating it.

Minor changes have been made to improve
reproduction quality.

Points of view or opinions stated in this docu-
ment do not necessarily represent official NIE
position or policy.

COMPUTER PROGRAMMING:

BASIC

Lincoln Public Schools
Lincoln, Nebraska
1980

"PERMISSION TO REPRODUCE THIS
MATERIAL HAS BEEN GRANTED BY

Jerry D. Beckman

Patience O. Fisher

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)."

"PERMISSION TO REPRODUCE THIS
MATERIAL HAS BEEN GRANTED BY

Charles D. Friesen

Fred H. Smith

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)."

ED225532

IR010534

TABLE OF CONTENTS

	<u>Page</u>
Introduction	1
General Goals of Computer Programming-BASIC	2
Specific Goals of Computer Programming-BASIC	3
BASIC Programming Commands and Concepts	12
Initializing a Diskette	13
System Commands	14
Editing Commands	18
Basic Commands	28
Disk Operating System (DOS)	33
Graphics (Low - Resolution)	45
Graphics (High - Resolution)	47
Traces	49
<u>Additional Resources</u>	
Supplemental Books	53

COMPUTER PROGRAMMING--BASIC

INTRODUCTION

For the past ten years the Lincoln Public Schools have offered to qualified high school students an introductory computer programming course in BASIC and/or FORTRAN. The general purpose of this course, Computer Programming, is to teach a programming language and to give students an opportunity to apply the language to solving problems. This course is not intended, however, to prepare skilled programmers who are ready for the job market.

In order to take Computer Programming a student must currently be enrolled in a mathematics course at or above the level of Algebra 3-4. The reason for this prerequisite is that until students attain this level of achievement in mathematics, they are unlikely to be able to deal with mathematical problems of such a degree of difficulty and sophistication as to require solution by a computer.

Until now students enrolled in Computer Programming in each high school have had access to a single teletype terminal. With the purchase of Apple II micro-computers and peripheral devices such as television monitors and disk drives, new possibilities have opened up. For example, more than one student at a time can now work with a computer. Also, because of the addition of television monitors, students can edit their programs with greater ease, and graphics can be added to the course. On the other hand, the substitution of micro-computers for the teletype terminals has meant the loss, at least temporarily, of the capability of teaching FORTRAN. Because of these changes, this guide has been prepared to assist those who teach the course to make the necessary adjustments.

The following persons participated in preparing this course guide:

Patience Fisher, Lincoln High
Charles Friesen, Lincoln High
Jerry Beckmann, Lincoln East
Fred Smith, Lincoln Southeast

June, 1980

GENERAL GOALS
of
COMPUTER PROGRAMMING--BASIC

The following goals are adopted from copyrighted material prepared by the Tri-County Goal Development Project. These goals may not be reproduced without permission from the Tri-County Goal Development Project Coordinator.

1. The student relates knowledge about computers to career goals and plans.
2. The student knows characteristics of computer-related hardware, software, and documentation, and principles underlying their design and use.
3. The student knows applications and uses of computers.
4. The student knows effects of computers on society.
5. The student knows principles, procedures and limitations of computer systems and can use computers as a tool for inquiry, problem-solving, and recreation.
6. The student is able to adapt concepts and processes of computer science to examine issues, to clarify personal values, to solve personal and social problems, and to satisfy personal curiosity.
7. The student knows and values ethical obligations and legal responsibilities related to computer use.
8. The student values knowledge and skills related to computing which enable individuals and groups to cope with the complexity of human society.
9. The student is able to make responsible decisions about computer uses that affect the student's economic, political, physical or social environment.

SPECIFIC GOALS
of
COMPUTER PROGRAMMING--BASIC
(Course Number 439)

The following goals are adopted from copyrighted material prepared by the Tri-County Goal Development Project. These goals may not be reproduced without permission from the Tri-County Goal Development Coordinator.

Goals marked with an asterisk (*) are major goals of this course.

Using Computers

- *1. The student is able to interact with a computer in learning situations.
- *2. The student is able to develop, code, and execute a program which produces valid output.
- *3. The student is able to read computer print-outs.
- *4. The student is able to select, interpret and use relevant data from computer print-outs when making decisions or solving problems.
- *5. The student knows causes of inaccuracies in the results from computer runs (e.g., round-off error, "garbage in/garbage out").
- *6. The student is able to generate test data to use in the comparison of alternate programs.

Problem Solving

- *7. The student is able to select appropriate procedures to be used at various stages of problem solving.
- *8. The student is able to determine the steps required in the solution of a problem.
- *9. The student is able to restate a problem in terms of its structural components (e.g., required information, necessary algorithm or tasks, desired output).
- 10. The student is able to select a problem solving method which takes into account ease of data entry and user convenience.
- *11. The student is able to use standard procedures to segment a complex problem into several smaller and more manageable problems.
- *12. The student knows that checking results for correctness is part of the problem-solving process.

Algorithms

- *13. The student knows that an algorithm is a definite sequence of steps to solve a particular problem within a finite time.
- *14. The student is able to select and/or develop an appropriate algorithm in problem-solving situations.
- *15. The student knows that more than one algorithm may exist to solve a problem.
- *16. The student is able to translate an algorithm into code.
- *17. The student is able to use a flow chart to represent an algorithm.
- *18. The student is able to correct errors in an improperly functioning algorithm.
- *19. The student is able to develop an algorithm that can be used to solve a set of similar problems.
- *20. The student is able to follow and give the correct output for a simple algorithm.
- *21. The student is able to explain a simple algorithm and what it accomplishes (i.e., interpret and generalize).
- *22. The student is able to modify a simple algorithm to accomplish a new, but related, task.
- *23. The student is able to detect logic errors in an algorithm.

Programming

- *24. The student knows that computer programs are sets of sequential instructions which accurately perform many problem solving activities (e.g., sorting, listing, calculating).
- *25. The student knows the syntax of BASIC.
- *26. The student knows basic programming procedures or steps (e.g., defining the problem, developing an algorithm, coding the program, documenting the program, locating and correcting errors).
- *27. The student knows the importance of logical structure.
- *28. The student is able to use programming procedures (e.g., loops, sub-routines, data structures).
- *29. The student knows functions and forms of branching statements (e.g., decision, iteration).
- *30. The student knows characteristics and function of an iterative process (e.g., counting, summing).
- *31. The student knows characteristics and purposes of subroutines and functions in BASIC.

- *32. The student knows characteristics and functions of arrays in BASIC.
- *33. The student is able to use input, output, data manipulation, arithmetic operations, and branching constructs.
- *34. The student is able to use programming techniques involving an iterative process (e.g., counters, summation).
- *35. The student is able to use programming techniques involving functions and subroutines.
- *36. The student is able to use arrays.
- *37. The student is able to analyze, construct, and evaluate arithmetic expressions using the hierarchy of operations of BASIC.
- *38. The student knows the forms and functions of constants, numeric variables and alphanumeric variables in BASIC.
- *39. The student is able to write a program to order information alphabetically or in numerical sequence.
- *40. The student is able to incorporate subroutines into a program being developed.
- *41. The student knows how control structures are implemented in BASIC (e.g., FOR-NEXT, IF-THEN).
- *42. The student knows functions of input and output statements.
- *43. The student is able to use appropriate BASIC statements to format input data.
- *44. The student is able to produce readable and informative output through the use of appropriate formatting.
- *45. The student is able to test a computer program for computational accuracy by using data with known output (e.g., textbook examples, verified calculations).
- *46. The student is able to design data to test various facets of a computer program (e.g., valid and invalid data to test all transfer of control statements, data to test output formatting).

Debugging

- *47. The student is able to use standard procedures to locate and correct logic errors in computer programs (e.g., intermediate output, computer tracing, manual tracing).
- *48. The student knows sources of errors in programs (e.g., round-off errors, overflow and underflow errors, syntax and logic errors).
- *49. The student is able to estimate for correctness of computer printouts.

- *50. The student is able to trace a program through its listing for the purpose of locating and correcting errors.
- *51. The student is able to trace a program to investigate program errors.
- *52. The student is able to use error messages and documentation to locate and correct syntax errors in a computer program.
- *53. The student is able to plan a strategy for tracing and correcting a computer-related error, such as a billing error.

Sorting

- *54. The student knows basic terms related to sorting, (e.g., ascending order, descending order).
- *55. The student knows an algorithm for sorting.
- *56. The student is able to write a computer program to sort a list of numbers.

Documentation

- *57. The student knows functions of documentation (e.g., assist in correcting errors, analyze an algorithm, modify a program, inform user of program capabilities and running procedures).
- *58. The student knows varieties of internal and external documentation.
- *59. The student knows the importance of adequately prepared internal and external documentation.
- 60. The student is able to make use of documentation when operating computing equipment.

Graphics

- *61. The student knows that computer graphics involves the use of a computer to generate and display drawings, graphs, and charts.
- 62. The student knows that computer systems can produce music and speech.
- *63. The student is able to use conventional steps in transforming a picture into computer output including deciding on the output design, drawing the picture on graph paper, deciding on a programming technique, and coding the program.
- *64. The student knows factors to consider when writing a program to generate graphical output (e.g., features of the language, output device, programming technique).
- 65. The student knows properties of transformations performed on a graphics display including translating, scaling and rotating.

Uses of Computers

66. The student knows uses of the computer as a tool to record, manipulate, store and transmit information, to order and analyze data, to stimulate intelligence, and to control machinery.
67. The student is able to recognize that the following activities are among the major types of applications of the computer:
- a. information storage and retrieval
 - b. simulation and modeling
 - c. process control--decision-making
 - d. computation
 - e. data processing
68. The student is able to recognize that computers are generally good at information-processing tasks that benefit from the following:
- a. speed
 - b. accuracy
 - c. repetition
 - d. mass data storage
 - e. complex computations
69. The student is able to recognize that some limiting considerations for using computers are as follows:
- a. cost effectiveness
 - b. software availability
 - c. storage capacity
 - d. time effectiveness
70. The student knows that a simulation usually attempts to build a model of the essential aspects of reality, so that reality may be better understood or controlled (e.g., testing effects of manipulating economic factors, testing the flight of an airplane wing before it is built).
71. The student knows typical employers of data processing personnel (e.g., banks, insurance companies, manufacturers of computer equipment, manufacturers of equipment containing computers, retail merchandizing firms, and data processing service organizations).
72. The student is able to determine how computers can be used to assist in a decision-making process.
73. The student is able to recognize specific uses of computers in some of the following fields:
- a. medicine
 - b. law enforcement
 - c. education
 - d. engineering
 - e. business
 - f. transportation
 - g. military defense systems
 - h. weather prediction
 - i. recreation
 - j. government
 - k. library
 - l. creative arts
74. The student is able to identify the fact that there are many programming languages suitable for a particular application for business or science.

75. The student knows types of numerical applications (e.g., linear programming, weather prediction, space shots, statistical analysis, mathematical proofs).
76. The student knows advantages and disadvantages of computer assisted instruction (e.g., advantages: providing immediate feedback, having infinite patience, keeping the learner actively involved, and individualized instruction; disadvantages: cost, possible lack of hardware or software to meet specific needs, inability to respond meaningfully under all circumstances).

Uses and Effects of Computers on Society

77. The student knows principal job classifications in the data processing field (e.g., computer operator, programmer, system analyst, systems programmer, computer center manager, and data preparation clerk).
78. The student knows employment projections in the data processing field.
79. The student knows examples of varied careers in which one may use the computer.
80. The student is able to distinguish among the following careers:
 - a. keypunch operator
 - b. computer operator
 - c. computer programmer
 - d. systems analyst
 - e. computer scientist
81. The student knows school courses which aid in preparation for data processing careers (e.g., math, introduction to business, typing, bookkeeping/accounting).
82. The student knows that as computers are used for more and more tasks, computer-related skills and experience become advantageous in an increasing variety of careers outside the computer industry and data processing fields.

Operation of the Computer

- *83. The student knows how a computer is accessed: for example
 - a. via a keyboard terminal
 - 1) at site of computer
 - 2) at any distance via telephone lines
 - b. via punched or marked cards
 - c. via other magnetic media (tape, diskette)
84. The student knows the basic operation of a computer system: input of data or information, processing of data or information, output of data or information.
85. The student recognized that alleged "computer mistakes" are usually mistakes made by people.
- *86. The student knows that data processing involves the transformation of data by means of a set of predefined rules.

- *87. The student knows that a computer needs instructions to operate.
- *88. The student knows that a computer gets instructions from a program written in a programming language.
- *89. The student knows that a computer is capable of storing a program and data.
- *90. The student knows that computers process data by searching, sorting, deleting, updating, summarizing, moving, and so on.
- 91. The student knows that information in digital computers is stored as electrical representations of zeros and ones.
- 92. The student knows that digital computers operated by a series of gates transmitting an on state or an off state. (Different computers implement these states different ways.)
- 93. The student recognizes that identification codes (numbers) and passwords are a primary means for restricting the use of computer systems, computer programs, and data files.
- *94. The student knows the difference between system commands and program statements (e.g., LIST, RUN vs. LET, GO TO, INPUT).
- *95. The student knows and is able to use the system commands for the specific system being used.
- 96. The student knows tasks typically done with library programs (simulations, business programs, math and science computations, computer assisted instruction, text editing, artistic creation).
- *97. The student is able to select, retrieve and execute computer programs.

Attitude, Values, and Motivation

- 98. The student does not feel fear, anxiety, or intimidation from computer experiences.
- *99. The student feels confident about his or her ability to use and control computers.
- 100. The student values economic benefits of computerization for a society.
- 101. The student enjoys and desires to work with computers.
- 102. The student describes past experiences with computers with positive-affect words, like fun, exciting, challenging, and so on.
- 103. The student spends some free time using a computer, when given an opportunity.

History

- 104. The student knows names and characteristics of early devices for manipulating numbers (e.g., Stonehenge, abacus, quipu, Napier's Bones, Pascal's adding machine, slide rule).
- 105. The student knows names and contributions of major figures in the historical development of mechanical computation (e.g., Napier, Jacquard, Pascal, Leibniz, Babbage, Hollerith, and Bush).
- 106. The student knows implications of John Van Neumann's contributions to the internal functions of the computer.
- 107. The student knows that early applications of the computer were largely numerical (e.g., artillery firing tables, solving sophisticated equations, aerospace industry).
- *108. The student is aware of the rapid growth of computer hardware since the 1940s.

Characteristics of Computer Systems

- 109. The student knows characteristics of various computer systems (e.g., real time, time-sharing, batch processing, interactive processing).
- 110. The student knows features and capabilities of microcomputer systems as compared to features and capabilities of large data processing systems.
- 111. The student knows how people may access computers, including: (a) via a keyboard terminal, at the site of the computer or at any distance via telephone lines; (b) via punched or marked cards; (c) via tape, diskette or other magnetic media.
- 112. The student knows meanings of terms and concepts basic to the understanding or use of computers (e.g., data, information, code, communication, library).

Peripheral Devices

- 113. The student knows characteristics and uses of common peripheral devices (e.g., hard copy terminal, cathode ray tube terminal, card reader, line printer, plotter, disk/tape drive).
- *114. The student is able to use appropriate peripheral devices for particular tasks.

Hardware and Software

- 115. The student knows that computer hardware refers to the physical components of computer systems.

- 116. The student is able to identify the five major components of a computer; input equipment, memory unit, control unit, arithmetic unit, output equipment.
- 117. The student knows that computer software refers to the programs required to use computer hardware (e.g., operating systems, compilers, interpreters, applications programs, file and data base programs).

Input-Output Devices

- *118. The student is able to use computer input-output devices to store, process, and display data.
- 119. The student can identify secondary storage systems (e.g., audio cassette, digital cassette, magnetic tape, floppy disk, hard disk).

Data

- 120. The student knows terms used by computer scientists to refer to individual units of computer storage (e.g., bit, byte, nibble, word).
- 121. The student knows distinguishing characteristics of different data types (e.g., integer data, floating point data, alphanumeric data).
- *122. The student knows characteristics of data structures, including simple variables, arrays, strings, lists, and files.
- 123. The student knows characteristics of sequential and random access files.
- 124. The student is able to store and retrieve data using text or random access files.
- *125. The student is able to use arrays for temporary storage and manipulation of data within programs (e.g., matrix functions).

BASIC Programming Commands and Concepts

Following is a list of the topics that should be covered in Computer Programming--BASIC. The order in which the topics are listed is not necessarily the order in which they should be taught.

BASIC Commands

PRINT (strings; numerical values)
GO TO: ON GO TO
END
LET (variable assignment; calculation type statements)
READ
DATA
RESTORE
FOR
NEXT
Math Functions (See Quick Reference Guide)
REM
INPUT
IF-THEN
DIMENSION
DEF (user defined functions)
GO SUB: RETURN
TAB: HTAB: VTAB: SPC

SYSTEM (Job Control) Commands

CATALOG	TRACE	SPEED =
LOAD	NO TRACE	HOME
SAVE	LIST	PR#1
NEW	DELETE	PR#0
RUN	CTRLS	INIT
		RESET

EDITING Commands

ESC	DEL
ESC A; B; C; D	CTRL X
Right Arrow; Left Arrow	

BASIC Concepts

Subscripted variables	Flow Charting
Initialization	Looping (with and without FOR-NEXT)
Counters	Stopping Procedures (Flags, Trailers, etc.)
Accumulators (summation)	Sorting
Use of commas; semicolons	Subscripted String Variables
Documentation	Subroutines

Evaluation Procedures

Quizzes; tests; programs; homework; major project; final exam

INITIALIZING A DISKETTE

Learning to use the disk and its operating system consists of learning a few special instructions. The first of these commands is INIT. This command is used to initialize a disk.

Use the following procedure to initialize your own disk. You need to do this so that you can store the programs that you will be writing. Perform the following steps:

1. Insert the System Master into the disk drive.
2. Turn on the monitor and the Apple II.
3. The red "in use" light will come on on the disk drive and the disk will make whirring and clacking noises. In a few seconds, a message will appear. This message may be similar to the following:

DISK II MASTER DISKETTE VERSION 3-2.1 PLUS

31-JULY-79

COPYRIGHT 1979 APPLE COMPUTER INC.

4. After the disk drive "in use" light goes off, remove the System Master from the drive and replace it with a blank diskette.
5. Type NEW, then type a greeting program. Here is a sample greeting program.

```
10 PRINT "THIS DISKETTE"  
20 PRINT "CREATED BY"  
30 PRINT "TERRY DOE"  
40 PRINT "JUNE 1980"  
50 END
```

You should supply your own name in place of Terry Doe and the current date in place of June 1980.

6. Type RUN. This will execute the program to see if it does what you expect.
7. Type INIT HELLO. When you press the RETURN key, the diskette will spin making clacks and whispery noises. The prompt character] will be displayed when initialization is complete.
8. Clear the monitor screen by typing HOME.
9. Type PR#6.
10. Type CATALOG. This will show you that the hello program is stored on your diskette.

SYSTEM (Job Control) COMMANDS

CATALOG

Displays on the screen the list of all files on the diskette.

LOAD

Example: LOAD RON

Attempt to find the program with a specified file named RON on the diskette. If found, that program will be LOADED into the memory of the computer.

SAVE

Example: SAVE RON

A file is created on the diskette and the current program is stored under the given file name, RON. If the diskette already contains a file with the specified file name, RON, the original file's contents are lost and the current program is saved in its place.

NEW

NEW command clears the computer's memory of any previous programs.

RUN

RUN causes the computer to begin execution of the current program.

LIST

LIST causes the complete program to be displayed on the screen.

LIST 200, 3000 causes program lines 200 through 3000 to be displayed.

LISTing is aborted by CTRL C.

To stop the LISTing temporarily at some point in the listing, use CTRL S.

Use CTRL S or the return key to resume the listing.

SYSTEM COMMANDS

CTRL C

- (a) Used to interrupt a running program or a listing.
- (b) Used to interrupt an INPUT if it is the first character entered.
The INPUT is not interrupted until the RETURN is pressed.

SPEED Example: SPEED = 200

Sets rate at which characters are to be sent to the screen. The slowest rate is 0 and the fastest is 255.

HOME

HOME is used to clear the text area and set the cursor to the top left corner of the currently defined text window.

PR#1

Transfers output to the printer.

PR#0

Returns output to the screen.

INIT

Refer to the initialization handout.

RESET

RESET stops any program or command unconditionally and immediately. This command leaves you in the system monitor program. To return to APPLESOFT without destroying the current stored program, type CTRL C RETURN.

APPLE SESSION (Illustrating System) COMMANDS

CATALOG

A 002 HELLO

] NEW

] HOME

] 10 PRINT "PLEASE TYPE IN YOUR NAME."

] 20 INPUT N\$

] 30 PRINT "PLEASE TYPE IN THE NAME OF YOUR SCHOOL."

] 40 INPUT S\$

] 50 PRINT N\$; "ATTENDS"; S\$

] 60 END

] LIST

10 PRINT "PLEASE TYPE IN YOUR NAME."

20 INPUT N\$

30 PRINT "PLEASE TYPE IN THE NAME OF YOUR SCHOOL."

40 INPUT S\$

50 PRINT N\$;" ATTENDS ";S\$

60 END

] RUN

PLEASE TYPE IN YOUR NAME.

?PAT SMITH

PLEASE TYPE IN THE NAME OF YOUR SCHOOL.

?XYZ HIGH SCHOOL

PAT SMITH ATTENDS XYZ HIGH SCHOOL

] SAVE PROGRAM 1

] CATALOG

A 002 HELLO

A 002 PROGRAM 1

] NEW

] LIST

] LOAD PROGRAM 1

] LIST

10 PRINT "PLEASE TYPE IN YOUR NA
ME."
20 INPUT N\$
30 PRINT "PLEASE TYPE IN THE NAM
E OF YOUR SCHOOL."
40 INPUT S\$
50 PRINT N\$;" ATTENDS ";S\$
60 END

] DELETE PROGRAM 1
] CATALOG

A 002 HELLO

] LOAD PROGRAM 1

FILE NOT FOUND

EDITING COMMANDS

A. ← Left Arrow Key

The left-arrow key, also called the backspace key, moves the cursor to the left. Each time the cursor moves to the left, one character is erased from the program line which you are currently typing, regardless of what the cursor is moving over. The screen is ignored by this command and nothing is changed on the screen.

B. → Right Arrow Key

The right-arrow key, also referred to as the retype key, moves the cursor to the right. As the cursor moves, each character it crosses on the screen is copied into Apple II's memory, just as if you had typed the character. It is used with the repeat key, to save re-typing an entire line when only minor changes are needed.

C. Repeat Key

The repeat key is labeled "REPT." If you hold down the repeat key while pressing a character key, the character will be repeated.

Pure Cursor Moves

The ESC, I, J, K, and M keys are used to move the cursor without affecting any of the characters on the screen.

<u>Command</u>	<u>Moves cursor</u>
ESC I	Up
ESC J	Left
ESC K	Right
ESC M	Down

Pressing ESC gets you into edit mode. Once you are in edit mode, pressing one of the above-mentioned letter keys will cause the cursor to move one character in the corresponding direction. For faster cursor motion, hold down one of these keys and then hold down the REPT key. The cursor will zip along while both keys are held down. To return to normal mode, press the space bar once.

EDITING COMMANDS

ESC Pressing ESC gets you into edit mode. Pressing the space bar gets you out of edit mode.

ESC is always used with another key

Pure Cursor Moves:

ESC K moves cursor to the right.

ESC J moves cursor to the left.

ESC M moves cursor down.

ESC I moves cursor up.

The above moves do not affect anything that is already displayed on the screen and does not affect anything in memory.

Right Arrow Key (Retype Key)

The right arrow key moves the cursor to the right and copies all characters and symbols it passes over into memory.

Left Arrow Key

The left arrow key moves the cursor to the left, erasing the character from memory as it passes over. The TV display is not changed when this key is used.

DEL

DEL x, y deletes lines x through y where x and y are program line numbers.

CTRL X

CTRL X will delete the line you are currently typing.

CTRL S

Stop Listing and resume listing.

Correcting Typing Errors

Abe B. Clod had a habit of printing the name of his favorite computer. Below is his program. As you can see, two words are misspelled.

```
10 PRINT "APPEL COMPUTOR"  
20 END
```

After listing the program, Abe corrected the typing errors as indicated below.

He typed: (Do not type commas!)

ESC, J, I, I, I

→ (repeatedly until cursor
is on the E)

L, E

→ (repeatedly until cursor
is on the second O)

E

→ (repeatedly, until cursor
is one or more spaces
past last quote)

Return

List

```
10 PRINT "APPLE COMPUTER"
```

```
20 END
```

to

Reason:

To place the cursor on the first
digit of the line number to be edited.

To retype everything up to the E.

Type an L first, then an E.

Retype up to the second O.

Replace the O with an E.

Retype R and ".

The line has been retyped.

Deleting One Line of a Program

```
10 PRINT "A"  
20 PRINT "B"  
30 PRINT "C"  
40 PRINT "D"  
50 END
```

]30

]LIST

```
10 PRINT "A"  
20 PRINT "B"  
40 PRINT "D"  
50 END
```

]

Deleting a Series of Consecutive Lines

1. Example One

```
10 PRINT "A"  
20 PRINT "B"  
30 PRINT "C"  
40 PRINT "D"  
50 END
```

]DEL 20, 40

]LIST

```
10 PRINT "A"  
50 END
```

]

You must use DEL (not DELETE!)

2. Example Two

LIST

```
10 PRINT "A"  
20 PRINT "B"  
30 PRINT "C"  
40 PRINT "D"  
50 END
```

]DEL 27, 43

]LIST

```
10 PRINT "A"  
20 PRINT "B"  
50 END
```

Since 27 is not an existing program line number, the next greater line number in the program is used instead of 27. Thus, 30 is actually used. On the other hand, since 43 is not an existing program line number, the next smaller line number is used. This means the 43 serves the same function as if it were 40. For this particular program, each of these statements would yield identical results:

DEL 27, 43

DEL 21, 49

DEL 30, 40

Deleting the Line You Are Currently Typing

Suppose you are typing a statement and suddenly realize you have made a mistake -- a mistake large enough that you would just as soon get rid of the entire line. If you press RETURN the statement will be stored in memory. By depressing the CTRL key and then the X key the line you are typing will be deleted. A "/" will appear on the screen when you type CTRL X. A sample Apple session below illustrates this situation.

```
10 PRINT "A"
]20 PRINT "I'VE BEEN WORKING ON THE/"
]30 PRINT "B"

]40 END

]LIST

10 PRINT "A"
30 PRINT "B"
40 END
```

By typing CTRL X immediately after the word "the," the symbol "/" will appear to indicate that the line has been deleted.

Inserting Text Into and Deleting Text From An Existing Line

Abraham B. Clod wished to type the first nine letters of the alphabet. He had entered this program to do the job.

```
10 PRINT "ABCGHI"  
20 END
```

Obviously, the letters D, E, and F were lost on the way to the forum. Abe's problem was to insert the D, E, and F between the C and the G. Here is how clever A. B. Clod did it. First he listed the program:

```
LIST  
10 PRINT "ABCGHI"  
20 END
```

Then he typed:

Reason:

ESC, J, I, I, I,	[This placed the cursor on the 1 in Line 10]
Space bar	[This takes us out of edit mode]
→ (until cursor on G)	[This retypes everything through the letter C]
ESC, I	[Back to edit mode and up one line to insert letters]
Space bar	[Get out of edit mode]
D, E, F	[Insert these letters immediately after C]
ESC, M, J, J, J	[Need to sneak back to the letter G]
Space bar	[Out of edit mode]
→ (repeat until cursor is one space beyond last quote)	[Retype from G to end of line]
Return	[The line has been edited]
LIST	
10 PRINT "ABCDEFGHI"	
20 END	
RUN	
ABCDEFGHI	

Inserting Text Into and Deleting Text From An Existing Line (continued)

Abe B. Clod was a perfectionist! He did not like to have ABCDEFGHI printed on the left margin of his video monitor. He wanted the printout to begin in the middle of the screen (i.e., on column 20). Thus, what he really wanted was for his program to look like this:

```
10 PRINT TAB (20); "ABCDEFGHI"
20 END
```

This is what A. B. Clod did:

He typed:

```
ESC, J, I, I, I
Space bar
→ (until on first quote)
ESC, I
Space bar
TAB (20);
ESC, M
J, J, J, J, J, J, J, J
Space bar
→ (until one space past
  last quote)
Return
LIST
10 PRINT TAB (20); "ABCDEFGHI"
20 END
RUN
      ABCDEFGHI
```

Because:

This placed the cursor on the I in line 10.
Out of edit mode.
Retypes everything up to first quote.
Edit mode and up one line to make insertion.
Out of edit mode.
Insert this expression before first quote.
Edit mode and down 1 line.
STAY IN EDIT MODE! Move cursor to first quote.
Out of edit mode.
Retype the statement from first quote to end of line.

Inserting Text Into and Deleting Text From An Existing Line (continued)

Mr. Clod was not really known as "Mr. Nimble Fingers," as the following program illustrates.

```
10 PRINT "ABCCDEEFF"  
20 PRINT "123489"  
30 END
```

What he really wanted was A, B, C, D, E, F, and 1, 2, 3, 4, 5, 6, 7, 8, 9.
He listed the program:

```
LIST  
10 PRINT "ABCCDEEFF"  
20 PRINT "123489"  
30 END
```

To make the necessary changes in the program he typed the following:

ESC, J, I, I, I, I (commas are not typed!)

Space bar

→ to first character to delete (C)

ESC, K,

Space bar

→ to next character(s) to delete (EF)

ESC, K, K

Space bar

→ to end of the line

RETURN

ESC J, I

Space bar

→ to 8

ESC M

Space bar

Inserting Text Into and Deleting Text From An Existing Line (continued)

5, 6, 7

ESC, I, J's to 8

→ to end of line

RETURN

LIST

RUN

Exercises

1. Use the edit commands to change the program below so that each digit will be printed exactly twice.

```
10 PRINT "112334455666778899"
```

```
20 END
```

The program should read as follows:

```
10 PRINT "112233445566778899"
```

```
20 END
```

DO NOT RETYPE THE PROGRAM! USE THE EDIT COMMANDS.

2. Use the edit commands to change the program below so that the two lines read:

```
DOESN'T TIME FLY
```

```
WHEN YOU'RE HAVING FUN!
```

```
10 PRINT "DOESN'T ME FLY"
```

```
20 PRINT "WHEEN YOU'RE HAVINGF!"
```

```
30 END
```

BASIC COMMANDS

PRINT

PRINT without any options causes line feed and return to be executed.

PRINT X causes the value of X to be printed followed by a line feed and return.

PRINT X, causes the value of X to be printed and the first character of the next item to be printed will be printed in the first position of the next available tab field. The first tab field comprises the left-most 16 printing positions (1 through 16). The second tab field occupies the next 16 positions (17 through 32) and the third tab field consists of the remaining 8 positions (33 through 40).

PRINT X; causes the value of X to be printed and the first character of the next item to be printed will be printed directly afterward with no intervening spaces.

PRINT A\$ causes the sequence of characters identified as A\$ to be printed.

GO TO

GO TO 200 causes the program to branch to line 200.

ON GO TO

ON 1 GO TO 100, 300, 400 causes the program to branch to line 100.

ON 2 GO TO 100, 300, 400 causes the program to branch to line 300.

ON 3 GO TO 100, 300, 400 causes the program to branch to line 400.

ON X GO TO 20, 10, 80 causes the program to branch to line 20 if X=1, line 10 if X=2 or line 80 if X=3.

GO SUB/RETURN

GO SUB 800 tells the computer to go to the subroutine beginning at line 800 and start executing at that statement. When a RETURN statement is executed, the program branches to the statement immediately following the most recently executed GO SUB.

END

END causes a program to cease execution and returns control of the computer to the user.

LET

LET A = X The variable name to the left of = is assigned the value of
LET B\$ = "ABC" the string or expression to the right of =.

READ

When the first READ statement is executed in a program, its first variable takes on the value of the first element in the DATA list. The DATA list consists of all the elements from all the DATA statements in the stored program. The second variable, if there is one, takes on the value of the second element in the DATA list, and so on. When the READ statement finishes execution, it leaves a data list pointer after the last element of data used. The next READ statement executed, if any, begins using the data list from the position pointer.

DATA

This statement creates a list of elements which can be used by READ statements. In order of instruction line number, each DATA statement adds its elements to the list of elements built up by the program's previous DATA statements.

RESTORE

This statement moves the data list pointer back to the beginning of the data list.

FOR/NEXT

Allows you to write a loop to perform a specified number of times any instructions between the FOR command (the top of the loop) and the NEXT command (the bottom of the loop).

```
FOR W = 1 to 20
:
NEXT W
```

In this example, the variable W counts how many times to do the instructions. The instructions inside the loop will be executed for W equal to 1, 2, 3, ---, 20, then the loop ends and the instruction after NEXT W is executed.

```
FOR Q = 2 to 20 STEP 2
```

```
NEXT Q
```

In this example Q will take on the values 2, 4, 6,...,20. Step 2 indicates that the variable Q is being incremented by 2.

Built-In Math Functions

- SIN (X) calculates the sine of X where X is in radians.
- COS (X) calculates the cosine of X where X is in radians.
- TAN (X) calculates the tangent of X where X is in radians.
- ATN (X) calculates the arctangent of X where X is in radians.
- INT (X) identifies the largest integer less than or equal to X.
- RND (X) identifies a random real number greater than or equal to 0 and less than 1. RND(0) identifies the most recently generated random number. If X is negative, RND(X) generates a particular random number that is the same every time. RND is used with that argument, and subsequent RND's with positive arguments will always follow a particular, repeatable sequence. If X is positive, a new random number from 0 to 1 is generated unless it is part of a sequence of random numbers initiated by a negative value for X.
- SGN (X) identifies a value of -1 if X is negative, 0 if X is 0 and 1 if X is positive.
- ABS (X) identifies the absolute value of X.
- SQR (X) identifies the positive square root of X. SQR executes more quickly than raising to the .5 power.
- EXP (X) raises e ($e=2.718289$) to the X power.
- LOG (X) identifies the natural logarithm of X.

REM

A REM statement is a non-executable statement. It is used to insert comments or notes into a program.

INPUT

INPUT X causes the computer to print a question mark and waits for the user to type a number, which will be assigned to the variable X.

INPUT "TYPE AGE THEN A COMMA THEN NAME"; B, C\$ causes the computer to print the string exactly as shown within the quotes, then waits for the user to type a number (which will be assigned to the variable B), then a comma, then string input (which will be assigned to the string variable C\$).

IF-THEN

If the expression following IF evaluates as true, then the instructions following THEN in the same line will be executed. Otherwise, any instructions following THEN are ignored, and execution passes to the instruction in the next numbered line of the program.

Examples: IF X < 18 THEN A = 0
IF A = "YES" THEN GO TO 80
IF R < X THEN GO TO 50

DIM

DIM is used to dimension subscripted variables. When a DIM statement is executed, it sets aside space for the specified array with subscripts ranging from 0 through the given subscript.

Example: DIM X(50) will cause the computer to reserve space in memory for 51 subscripted variables X(I) where I ranges from 0 to 50.

If an array element is used in a program before it is DIMensioned, a maximum subscript of 10 is allotted for each dimension in the element's subscript.

DEF (user defined functions)

DEF allows the user to define functions in a program. First the Function FN name is defined using DEF. Once the program line DEFining the function has been executed, the function may be used in the form FNname (argument) where the argument may be any arithmetic expression.

Example: 10 DEF FN A(W) = 2 x W + W
20 PRINT FN A(23)

TAB

TAB(X) moves the cursor to the position that is X printing positions from the left margin of the text window if X is greater than the value of the current cursor position relative to the left margin. If X is less than the value of the current cursor position, then the cursor is not moved. TAB never moves the cursor to the left. TAB must be used in a print statement.

HTAB

HTAB X moves the cursor to the position that is X positions from the left edge of the current screen line. HTAB's moves are relative to the left margin of the text window, but independent of the line width. Assume the line in which the cursor is located has positions 1 through 255. Regardless of the text window width you may have set, positions 1 through 40 are on the current line, positions 41 through 80 are on the next line down, and so on.

HTAB 1 moves the cursor to the leftmost position of the current line.

HTAB 0 moves the cursor to position 256.

VTAB

VTAB Y moves the cursor to the line that is Y lines down on the screen. The top line is line 1 and the bottom line is line 24. This statement may involve moving the cursor either up or down, but never to the right or left.

SPC

SPC (X) introduces X spaces between the items previously printed and the next item to be printed. X must be between 0 and 255 inclusive. The SPC command must be used in a PRINT statement.

DISK OPERATING SYSTEM (DOS)

Very often it's useful to use a DOS command from within a BASIC program. Included are these Housekeeping commands:

INIT	RUN	LOCK	MAXFILES
CATALOG	RENAME	UNLOCK	MON
SAVE	DELETE	VERIFY	NOMON
LOAD			

These Sequential Text File Commands are also DOS commands.

OPEN	APPEND
CLOSE	POSITION
READ	EXEC
WRITE	

Sequential Text Files

At some time you may want to use the disk to store information that is not a program. For example, you might wish to store a series of numbers or a list of words. Later you may wish to retrieve these numbers or words and "do something" with them. In the case of numbers, you might want to sum them. In the case of words, alphabetizing might be the appropriate operation. A text file (sometimes called a data file) will allow you to do these tasks. The letter T marks text files in the CATALOG directory.

```
*A 010 RANDOM
(*T)003 APPLE PROMS
*A 039 RENUMBER INSTRUCTIONS
*A 014 RENUMBER
*A 006 BRIAN'S THEME
*A 058 HOPALONG CASSIDY
```

Sequential text files should be used when information is to be retrieved in a linear fashion from the beginning to the end of the file, and when information does not require much updating or on-going revision.

To create a sequential text file, the commands

```
OPEN
WRITE
PRINT
CLOSE
```

are used, in the order shown (though not necessarily right after each other).

To retrieve a sequential text file, the commands

OPEN
READ
INPUT
CLOSE

are used, again in the order shown, though not necessarily right after each other.

A certain ritual is required before and after you create (WRITE) a sequential text file: before using the file you must OPEN it. When you are done, you must CLOSE it. When retrieving (READING) a sequential text file: OPEN before READING, and CLOSE (quietly please) the file when you are done.

Files that have been OPENed must be CLOSEd. Failure to CLOSE a file that was OPENed and written to by a WRITE command may result in loss of data.

An example of a program which creates a sequential text file is given below. A catalog listing of the programs on the diskette is first given. Note: the text file "DATA" is not found in the list of stored programs. However, it will be listed after TEXTMAKER has been RUN. By running TEXTMAKER, the sequential text file DATA will be created.

CATALOG

*A 002 HELLO
A 010 THREE DIM PLOT
A 007 ANIMATION
A 022 POLYGRAPHS
A 002 TEXTMAKER
A 002 TEXTRETRIEVER

]LOAD TEXTMAKER
]LIST

```
10 FOR X = 1 TO 5
20 INPUT Z (X)
30 NEXT X
40 D$ = CHR$ (4)
50 PRINT D$; "OPEN DATA"
60 PRINT D$; "WRITE DATA"
70 FOR X = 1 TO 5
80 PRINT Z (X)
90 NEXT X
100 PRINT D$; "CLOSE DATA"
110 END
```

]RUN
?1
?2
?3
?4
?5

Lines 10, 20, and 30 will ask for five numbers to be entered. The numbers 1, 2, 3, 4, 5 were entered following the question marks.

D\$ is assigned CHR\$(4). In order to use a DOS command from within a BASIC program, this assignment must be made.

Line 50 opens the sequential text file called DATA.

Line 60's WRITE command causes subsequent output from PRINT statements to be sent to the text file named DATA. So in the program above, each execution of line 80 will send a number to the text file DATA.

Line 100 politely CLOSEs the file and ends the file-writing process.

After executing TEXTMAKER, a CATALOG listing shows that the text (T) file DATA now exists.

]CATALOG

```
*A 002 HELLO
A 010 THREE DIM PLOT
A 007 ANIMATION
A 022 POLYGRAPHS
T 002 DATA
A 002 TEXTMAKER
A 002 TEXTRETRIEVER
```

]NEW

An example of a program (TEXTRETRIEVER) which will retrieve the sequential text file DATA is given below.

]LOAD TEXTRETRIEVER
]LIST

```
10 D$ = CHR$ (4)
20 PRINT D$; "OPEN DATA"
30 PRINT D$; "READ DATA"
40 FOR X = 1 TO 5
50 INPUT Z (X)
60 PRINT Z (X)
70 NEXT X
80 PRINT D$; "CLOSE DATA"
90 END
```

]RUN

1
2
3
4
5

Line 20 opens the file.

Line 30 tells DOS that all subsequent INPUT (or GET) statements will refer to the file DATA instead of Apple's keyboard. The disk will supply the responses to the INPUT statements (Line 50), instead of you.

Line 50 will be executed five times. The five numbers stored in DATA will be assigned to Z (1), Z (2), ... Z (5).

Line 60 will cause the number just retrieved from DATA to be printed.

Line 80 closes the file named DATA.

Suppose that we now wish to sum the five numbers stored in DATA. The program below will read and print the five numbers retrieved from DATA. The sum will then be printed.

```
10 D$ = CHR$ (4)
20 PRINT D$; "OPEN DATA"
30 PRINT D$; "READ DATA"
40 FOR X = 1 TO 5
50 INPUT Z (X)
60 PRINT Z (X)
65 S = S + Z (X)
70 NEXT X
80 PRINT D$; "CLOSE DATA"
85 PRINT "SUM = ";S
90 END
```

]RUN

1

2

3

4

5

SUM = 15

Note: Line 85 may follow the CLOSING of the DATA file. The values assigned to S, Z (1), Z (2), Z (3), Z (4), and Z (5) are being stored in the memory of the Apple. Thus, one can use these variables and their values after the file has been closed.

The next problem involves READING the values in DATA. The values will be printed in the order in which they are stored. After the sequential text file DATA is CLOSED, the numbers will be printed in reverse order. Again, note that this last task was performed after the file had been CLOSED.

```

10 D$ = CHR$ (4)
20 PRINT D$; "OPEN DATA"
30 PRINT D$; "READ DATA"
40 FOR X = 1 TO 5
50 INPUT Z (X)
60 PRINT Z (X)
70 NEXT X
80 PRINT D$; "CLOSE DATA"
90 PRINT
100 PRINT "NOW WE WILL REVERSE THE ORDER"
110 PRINT
120 FOR X = 5 TO 1 STEP - 1
130 PRINT Z (X)
140 NEXT X
150 END

```

]RUN

1
2
3
4
5

NOW WE WILL REVERSE THE ORDER

5
4
3
2
1

RANDOM-ACCESS FILES

Random-Access text files stores information in "cells" called records. Random-Access files should be used in situations requiring fast access to various parts of the file, or where individual pieces of information in the file need to be changed quite often.

Random-Access files are created and retrieved in a manner very similar to that used for sequential files, the main difference is that certain commands have additional parameters.

OPEN (required an L; length of record parameter)

READ] (requires an R; record number parameter)
WRITE]

Let's create a random-access file called RDATA and examine the syntax for the above Commands.

OPEN RDATA, L(j) (where (j) indicates how many characters and digits are to be allotted to each record in the file)

CLOSE RDATA (the close Command works exactly as it did for sequential files)

WRITE RDATA, R(j)

READ RDATA, R(j) where (j) specifies the record number (i.e, the 1st, 2nd, 3rd, etc.)

You will have to keep detailed written documentation on the structure and contents of your files. It might behelpful to always include the Length parameter in each file's name. (RDATA-L5)

An example of a program (RANDOMMAKER) which creates a random-access text file is given below:

```
5  REM : RANDOMMAKER
10 DS = CHR$ (4)
11 FOR X = 1 to 5
12 A(X) = X
13 NEXT X
20 PRINT DS; "OPEN RDATA, L5"
30 FOR I = 1 TO 5
40 PRINT DS; "WRITE RDATA, R"; I
50 PRINT A(I)
60 NEXT I
70 PRINT DS; "CLOSE RDATA"
80 END

]RUN
```

Line 11, 12 and 13 assigns the numbers 1 to 5 for X(1), X(2), X(3), X(4), and X(5).

Line 20 opens the file RDATA with record length of 5.

Each execution of lines 30, 40, 50 and 60 will send a number to the random text file RDATA.

Line 40 will store each number in a different record.

An example of a program (RANDOM Retriever) which will retrieve the random file RDATA is given below.

```
5  REM :RANDOMRETRIEVER
10 D$ = CHR$ (4)
20 PRINT D$; "OPEN RDATA,L5"
30 FOR I = 1 to 5
40 PRINT D$; "READ RDATA,R";I
50 INPUT " ";X
55 PRINT X
60 NEXT I
70 PRINT D$;"CLOSE RDATA"
80 END

]RUN
1
2
3
4
5
```

Line 20 opens the file

Line 40 tells DOS that all Input statements will refer to the file RDATA instead of the Apple Keyboard. The disk will supply the responses instead of you.

Line 50 will be executed 5 times. The five numbers stored in RDATA will be assigned to X(1), X(2), X(3), X(4) and X(5).

Line 55 will cause the number just retrieved from RDATA to be printed.

An example of a program that retrieves only the numbers stored in the first and third records is given below:

```
5 REM :RANDOMRETRIEVER
10 D$ = CHR$ (4)
20 PRINT D$;"OPEN RDATA,L6"
30 FOR I = 1 to 3 STEP 2
40 PRINT D$;"READ RDATA,R";I
50 INPUT " ";X
55 PRINT X
60 NEXT I
70 PRINT D$;"CLOSE RDATA"
80 END

]RUN
1
3
```

The following programs illustrate the storage and retrieval of words placed in a random text file.

```
5 REM :RANDOMMAKER
10 D$ = CHR$ (4)
11 FOR X = 1 to 5
12 INPUT A$(I)
13 NEXT X
20 PRINT D$;"OPEN RDATA,L6"
30 FOR I = 1 to 5
40 PRINT D$;"WRITE RDATA,R";I
50 PRINT A$(I)
60 NEXT I
70 PRINT D$;"CLOSE RDATA"
80 END

]RUN
?ONE
?TWO
?THREE
?FOUR
?FIVE
```

Line 12 will require you to enter the words to be placed in the file.
These words are entered following the question marks during the run of
the program.

```
5  REM :RANDOM RETRIEVER
10 D$ - CHR$ (4)
20 PRINT D$;"OPEN RDATA,L6"
30 FOR I = 1 to 5
40 PRINT D$;"READ RDATA,R";I
50 INPUT " ";X$
55 PRINT X$
60 NEXT I
70 PRINT D$;"CLOSE RDATA"
80 END
```

]RUN

ONE
TWC
THREE
FOUR
FIVE

POSITION

The DOS command POSITION will allow you to access information from any specified field within a text file. The statement below illustrates how the POSITION command might appear in a program.

```
PRINT D$; "POSITION DATA, R2"
```

You will recall that the sequential text file named DATA contained five numbers (1, 2, 3, 4, and 5). The R2 is the Relative-field position. This command specifies that DOS's position-in-the-file pointer will move forward to the 2nd field ahead of the current pointer position. R0 would indicate beginning with the current field. R1 would suggest skipping the current field and beginning in the next field.

POSITION must refer to a file that you have already OPENed. OPEN automatically sets the position-in-the-file pointer back to the beginning of the first field. POSITION must be used before the associated READ or WRITE.

The program below will read and print the third field in the file DATA. (Note: R3 would have set the pointer at the 4th field!)

```
10 REM TEXTPOSITION
20 D$ = CHR$ (4)
30 PRINT D$; "OPEN DATA"
40 PRINT D$; "POSITION DATA,R2"
50 PRINT D$; "READ DATA"
60 INPUT A
70 PRINT A
80 PRINT D$; "CLOSE DATA"
90 END
```

```
]RUN
3
]
```

The next program will first assign X to the value 1, and then the value 3. Thus, statement 40 will actually read

```
40 PRINT D$; "POSITION DATA, R1" the first time through the for-next
loop. The second time it will be
40 PRINT D$; "POSITION DATA, R3".
```

In this program the 2nd and 4th fields will be read (statements 50 and 60) and printed (statement 70). In addition, these numbers will be summed (statement 65). The average of the two numbers will also be printed (statement 87).

```
10 REM TEXTPOSITION
20 D$ = CHR$ (4)
25 FOR X = 1 TO 3 STEP 2
30 PRINT D$; "OPEN DATA"
40 PRINT D$; "POSITION DATA,R";X
50 PRINT D$; "READ DATA"
60 INPUT A
65 S = S + A
70 PRINT A
80 PRINT D$; "CLOSE DATA"
85 NEXT X
87 PRINT "THE AVERAGE IS"; S / 2
90 END
```

]RUN

2

4

THE AVERAGE IS 3

APPEND

The DOS Command APPEND will allow you to add text to the end of a sequential text file. The file named DATA contains five fields, namely the numbers 1, 2, 3, 4, 5. Suppose we wish to add the words "six" and "seven" to this file. The program below will do this through the use of the APPEND Command.

```
5 REM : APPEND DATA
10 D$ = CHR$ (4)
20 PRINT D$; "APPEND DATA"
30 PRINT D$; "WRITE DATA"
40 PRINT "SIX"
50 PRINT "SEVEN"
60 PRINT D$; "CLOSE DATA"
70 END
```

The program on the following page will OPEN DATA. It then calls for the reading and printing of five numerical values (statements 40-70). Following this, two string variables are read and printed (statements 71-74). The printout confirms the fact that DATA contains seven fields and that the APPEND Command used in the previous program was used successfully.

```
10 D$ = CHR$ (4)
20 PRINT D$; "OPEN DATA"
30 PRINT D$; "READ DATA"
40 FOR X = 1 TO 5
50 INPUT Z (X)
60 PRINT Z (X)
70 NEXT X
71 FOR X = 1 TO 2
72 INPUT X$
73 PRINT X$
74 NEXT X
80 PRINT D$; "CLOSE DATA"
90 END
```

```
]RUN
1
2
3
4
5
SIX
SEVEN
```

GRAPHICS

Low - Resolution

In graphics mode, the Apple displays up to 1600 small squares (also called dots or points) on a 40 by 40 grid. Each square is assigned a location by using an ordered pair. The point (10, 30) would be located in the 10th column and 30th row. In general the point (x, y) would be in the xth column and yth row. The square in the upper left corner is assigned (0, 0). All other points are located with (0, 0) as the reference point.

Six commands are needed to work with low resolution graphics. They are:

GR	Sets the graphics mode.
Color = 15	Assigns a white line or point.
Plot	Places a dot at the specified location.
HLIN	Draws a horizontal line at the specified location.
VLIN	Draws a vertical line at the specified location.
TEXT	Sets the screen to the usual non-graphics text mode.

Do the following steps on the Apple

```
]GR
]COLOR = 15
]PLOT 10, 10    Places a point in 10th column and 10th row.
]HLIN 0, 39 AT 20    Draws a horizontal line from column 0 to column 39
                    in the 20th row from the top.
]VLIN 0, 39 AT 20    Draws a vertical line from row 0 to row 39 in the
                    20th column from the left.
]TEXT
```

Now type in the following two programs.

NEW

```
10 GR
20 COLOR= 15
30 PLOT 0, 0
40 PLOT 0, 39
50 PLOT 39, 39
60 PLOT 39, 0
70 END
```

This program places a point in each corner of the grid.

]RUN

]NEW

]10 GR

]20 COLOR= 15

]30 HLIN 0, 39 AT 20

]40 VLIN 0, 39 AT 20

]50 END

]RUN

]TEXT

This program produces a horizontal line and a vertical line which intersect at the center of the grid.

Exercises

1. Make a square and place a point in the middle.
2. Make a 3 by 3 grid (tic-tac-toe board) and put a point in the middle of each square.
3. Use a piece of graph paper to sketch out the 3 letters of your school on a 40 by 40 grid. Write a program to print this on the computer. Make each letter 20 spaces high.

GRAPHICS

High - Resolution

The commands of high-resolution graphics are very similar to what you did with low-resolution graphics. They are usually formed by placing an H in front of the command you already know.

In High-resolution graphics mode, the Apple permits you to plot points on a grid 280 positions wide by 160 positions high.

Three commands are needed to work with high-resolution graphics. They are:

HGR	Sets high-resolution graphics mode.
HPOINT x, y	Places a dot at the specified location x, y. (x is from 0 to 279; y is from 0 to 159).
HPOINT x1, y1 to x2, y2	Draws a line from the point (x1, y1) to the point (x2, y2). This can be extended to additional points.

Do the following steps on the Apple.

```
]HGR
[HPOINT 10, 20 TO 40, 100 ----- Draws a line from the point (10,20) to the
                                     point (40, 100).
]HGR
[HPOINT 5,100 TO 50,50 TO 80,100 -- Draws a broken line from the points (5,100)
                                     to the point (50,50) to the point (80,100).
]TEXT
```

Now type in the following program.

```
]NEW
]10 HGR
]20 HPOINT 5,10 TO 5,120
]30 HPOINT 5,65 TO 40,65      This program graphs a message.
]40 HPOINT 40,10 TO 40,120
]50 HPOINT 80,10 TO 80,120
]60 END
]RUN
```

Exercises

1. Draw a 200 x 100 rectangle and put in the diagonals.
2. Draw a 4 x 4 square game board and put in all the diagonals.
3. Graph your name. (Use block letters.)

TRACES

To see the path of execution in a program, you can invoke a special command (TRACE). The TRACE command can be issued either in immediate mode or as a program statement. In either mode, once the TRACE command is issued, your program will be TRACEd every time you RUN it. To stop TRACE, you must issue a NOTRACE command, either in a line of your program or in immediate mode.

```
10 PRINT "THIS PROGRAM WILL"
11 PRINT "ILLUSTRATE THE USE OF"
12 PRINT "THE TRACE COMMAND"
20 X = 1
30 GOTO 80
40 Y = 2
50 GOTO 100
60 Z = 3
70 GOTO 120
80 PRINT X
90 GOTO 40
100 PRINT Y
110 GOTO 60
120 PRINT Z
130 END
```

```
]RUN
THIS PROGRAM WILL
ILLUSTRATE THE USE OF
THE TRACE COMMAND
1
2
3
```

]TRACE ----- Immediate mode use of TRACE.

```
]RUN
#10 THIS PROGRAM WILL
#11 ILLUSTRATE THE USE OF
#12 THE TRACE COMMAND
#20 #30 #80 1
#90 #40 #50 #100 2
#110 #60 #70 #120 3
#130
```

]NOTRACE ----- Immediate mode use of NOTRACE.

Program mode use of TRACE and NOTRACE.

```
10 PRINT "THIS PROGRAM WILL"
11 PRINT "ILLUSTRATE THE USE OF"
12 PRINT "THE TRACE COMMAND"
19 TRACE
20 X = 1
30 GOTO 80
40 Y = 2
50 GOTO 100
60 Z = 3
61 TRACE
70 GOTO 120
80 PRINT X
81 NOTRACE
90 GOTO 40
100 PRINT Y
110 GOTO 60
120 PRINT Z
130 END
```

```
]RUN
THIS PROGRAM WILL
ILLUSTRATE THE USE OF
THE TRACE COMMAND
#20, #30, #80 1
#81 2
#70 #120 3
#130
]
```

In line 19, the TRACE command is issued. From this point on, each line number of the lines being executed will be displayed until the NOTRACE is issued in line 81. From line 81 on no line numbers will be displayed until line 61 when the TRACE command is again issued. Compare the trace of the program mode with the trace of the immediate mode.

The program on the following page contains a logic error. The program would run forever because of the infinite loop that was created. To stop the printing, type CTRL C. This will result in BREAK IN to be displayed. The TRACE command in immediate mode is now issued to display the flow of execution by running the program.

```

10 X = 1
20 Y = 2
30 PRINT Y,
40 Y = Y + 2
50 IF Y < 10 THEN 30
55 PRINT
56 PRINT
57 PRINT
60 X = X + 1
70 IF X > 3 THEN 90
80 GOTO 10
90 END

```

]RUN

2	4	6
8		

2	4	6
8		

2	4	6
8		

2	4	6
8		

2	4	6
8		

2	4	6
8		

BREAK IN 55
]TRACE

]RUN

#10	#20	#30	2	#40	#50	#30	4
#40	#50	#30	6	#40	#50	#30	8
#40	#50	#55					
#56							
#57							
#60	#70	#80	#10	#20	#30	2	
#40	#50	#30	4	#40	#50	#30	6
#40	#50	#30	8	#40	#50	#55	
#56							
#57							

```

#60 #70 #80 #10 #20 #30 2
#40 #50 #30 4   #40 #50 #30 6
#40 #50 #30 8   #40 #50 #55
#56
#57
#60 #70 #80 #10 #20 #30 2
#40 #50 #30 4   #40 #50 #30 6
#40 #50
BREAK IN 50
]

```

In the following program, the TRACE command is used to illustrate the flow of execution in a more complex program.

```
]250 END
```

```
]LIST
```

```

5 Y = 1
10 FOR X = 1 TO 5
20 PRINT X
30 NEXT X
40 GOTO 200
50 IF Y > 3 THEN 200
60 PRINT "THE END"
70 GOTO 250
200 FOR X = 5 TO 1 STEP - 1
210 PRINT X
220 NEXT X
230 Y = Y + 1
240 GOTO 50
250 END

```

```
]RUN
```

```

1
2
3
4
5
5
4
3
2
1
5
4
3
2
1
THE END

```

```
]TRACE
```

```
]RUN
```

```

#5 #10 #20 1
#30 #20 2
#30 #20 3
#30 #20 4
#30 #20 5
#30 #40 #200 #210 5
#220 #210 4
#220 #210 3
#220 #210 2
#220 #210 1
#220 #230 #240 #50 #200 #210 5
#220 #210 4
#220 #210 3
#220 #210 2
#220 #210 1
#220 #230 #240 #50 #60 THE END
#70 #250

```

55

SUPPLEMENTAL BOOKS

Ahl, David H., ed. The Best of Creative Computing, Volume 2. Morristown, N.J., Creative Computing Press, 1977.

_____, ed. The Best of Creative Computing, Volume 1, Morristown, N.J. Creative Computing Press. 1976.

_____, ed. 101 Basic Computer Games, Maynard, Mass. Digital Equipment Corp., 1973.

Albrecht, Robert. My Computer Likes Me When I Speak Basic. Menlo Park, CA. Dymax, 1972.

Bent, Robert J. and Sethares, George C. Basic An Introduction to Computer Programming. Monterey, CA., Brooks/Cole Publishing Company, 1978.

Calter, Paul. Problem Solving with Computers, McGraw-Hill Book Company, 1973.

Coan, James S. BASIC, BASIC. Rochelle Park, New Jersey, Hayden Book Company, Inc., 1970.

Corliss, William A. Computers, United States Energy Research and Development Administration, Washington, D.C.

Chou, George Tsu der. Computer Programming in BASIC. Canfield Press, San Francisco, Calif. 1979.

De Rossi, Claude J. Learning Basic Fast. Reston, Virginia, Reston Publishing Co., Inc. 1974.

Doerr. Micro-Computers in the Schools. Rochelle Park, N.J. Hayden Book Co., 1979.

Dwyer, Thomas, A Guided Tour of Computer Programming in BASIC. Boston, Mass., Houghton Mifflin Company, 1973.

Dwyer, Thomas and Margot Critchfield., Computer Resource Book - Algebra., Boston, Mass., Houghton Mifflin Co., 1975.

- Edwards, Perry and Broadwell, Bruce, Flowcharting and Basic. Harcourt Brace-Jovanovich, Inc., 1974.
- Frederick, Janet, et al., Conduit BASIC Guide., Iowa City, Iowa, the University of Iowa, 1979.
- Golden, Neal. Computer Programming In The Basic Language. Harcourt-Brace-Jovanovich, 1975.
- Gruenberger, Fred and Jaffray, George. Problems For Computer Solution. John Wiley & Sons, Inc. 1965.
- Hamming, Richard W. Calculus and the Computer Revolution. Boston, Mass., Houghton-Mifflin, Company, 1968.
- Hennefeld, Julien, Using BASIC: An Introduction to Computer Programming., Boston, Mass., Prindle Weber and Schmidt, 1978.
- Jacobs, Zeney, et al. Computer Programming in the BASIC Language. Boston, Mass., Allyn and Bacon, 1978.
- Katzman, Pamela W. Computer Assisted Mathematics Program - Geometry. Scott, Foresman and Company, 1970.
- Kieren, Thomas. Computer Assisted Mathematics Program Intermediate Mathematics. Glenview, Illinois., Scott, Foresman and Company, 1969.
- LaFrenz, Dale E. Computer Assisted Mathematics Program - Algebra. Scott, Foresman and Company, 1969.
- Ledin, George Jr. A Structured Approach to General BASIC., San Francisco, CA., Boyd & Fraser Publishing Co., 1978.
- Logsdon, Tom, Programming in BASIC with Applications. Fullerton, CA., Anaheim Publishing Co., 1977.
- Logsdon, Tom. Programming in BASIC. Fullerton, Calif. Anaheim Publishing Company, 1978.
- Marateck, Samuel L. BASIC, New York, Academic Press, 1975.
- McCunn. Computer Programming for the Complete Idiot. San Francisco, Calif. Design Enterprises of San Francisco, 1979.

Moody, The First Book of Micro-Computers, Rochelle Park, N.J. Hayden Book, Co.
1978.

Moursund, David. Basic Programming for Computer Literacy. New York: McGraw-Hill,
1978.

NCTM., Introduction to an Algorithmic Language (BASIC). Washington, D.C., NCTM., 1968.

Pavlovich, Joseph P. and Taham, Thomas E., Computer Programming In Basic,
Holden-Day, 1971.

Peckham, Herbert D., BASIC: A Hands-on Method., New York, N.Y. McGraw-Hill, Book,
Comp. 1978.

Pegels, Carl C. BASIC: A Computer Programming Language with Business and Management
Applications., Holden-Day, 1973.

Rogowski, Stephen J., Problems for Computer Solution. Hartford, Conn. EDUCOMP,
Corporation., 1975.

Sack, John and Meadows, Judith. Entering Basic, Science Research Assoc., Inc.,
1973.

Sage, Edwin R. Problem-Solving With the Computer, Newburyport, Mass. ENTELEK,
Incorporated., 1969.

Sharpe, William F. and Nancy L. Jacob., BASIC: An Introduction to Computer Programming
Using the BASIC Language. The Free Press, New York, N.Y. 1979.

Simon, David E. BASIC From the Ground Up, Rochelle Park, New Jersey, Hayden Book
Company, Inc., 1978.

Smith, Robert E. Discovering BASIC A Problem Solving Approach., New York, Hayden
Book Company, Inc., 1970.

Spencer. Sixty Challenging Problems With BASIC Solutions, 2nd. Ed. Rochelle Park,
N.J., Hayden Book Co., 1979.

Squire. The Computer: An Everyday Machine, 2nd. Ed. Reading, MA, Addison, Wesley.

Traeton, Ken. 57 Practical Programs & Games in Basic. Tab Books, 1978.

Weik. Standard Dictionary of Computers and Information Processing., 2nd ed.,
Rochelle Park, N.J. Hayden Book, Co., 1977.

Worland, Peter, Introduction to BASIC Programming A Structured Approach., Houghton-
Mifflin, Company, Boston, Mass. 1979.

Worth, Thomas., BASIC For Everyone, Prentice-Hall, Inc., Englewood Cliffs, N.J. 1976.

Wu, Nesa L'abbe, BASIC: The Time-Sharing Language., Dubuque, Iowa, Wm. C. Brown
Company Publishers, 1975.

PERIODICALS

Byte, Byte Publications, Inc., 70 Main, Peterborough, N.H. 03458
Monthly, \$18.00

Creative Computing, P.O. Box 789-M, Morristown, N.J., 07412
Monthly, \$15.00

Kiloband, Monthly. \$23.00

On Computing, P.O. Box 307, Martinsville, N.J., 08836
Quarterly

Personal Computing, Benevill Pub. Co., 167 Corey Road, Brookline, MA, 02146
Monthly, \$22.00

Recreational Computing, People's Computer Co., 1263 El Camino Real, Box E,
Bi-Monthly, \$10.00 Menlo Park, CA 94025

The Computing Teacher, Oregon Council for Computer Education, Computing
Center, Eastern Oregon State College, LaGrange, OR 97850

OTHER RESOURCES

Adventure of the Mind (ETV - Six 15-minute programs)

TIES Bibliography of Instructional Resources